VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Nástroj pro Online Průzkumy s podporou statistického vyhodnocování

Online Survey Tool with Statistical Evaluation Support

2013 **Michal Zelenka**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Michal Zelenka**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Nástroj pro Online Průzkumy s podporou statistického vyhodnocování
Online Survey Tool with Statistical Evaluation Support

Zásady pro vypracování:

Postup pro vypracování (specifické cíle práce):

1. Prostudujte problematiku online dotazníků a existující nástroje na internetu.
2. Navrhněte webovou aplikaci pro podporu eletronických průzkumů.
3. Detailně zpracujte specifikaci a samotný návrh řešení.
4. Uvedený návrh naimplementujte v prostředí Java EE s pomocí vhodného webového frameworku a výsledný produkt otestujte.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Ježek, Ph.D.**

Datum zadání: 18.11.2011

Datum odevzdání: 18.07.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení Studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: 17.7.2014

Zelenka.....

Podpis studenta

Poděkování

Rád bych poděkoval svému vedoucímu práce Ing. Oldřichu Vašutovi, za cenné rady a připomínky, které mi během zpracování mé bakalářské práce poskytoval. A také bych rád poděkoval svému druhému vedoucímu Ing. Davidu Ježkovi Ph.D., který mi byl oporou při opravě této bakalářské práce.

Abstrakt

Hlavní cílem této bakalářské práce je navrhnout webovou aplikaci pro online průzkumy s podporou statistického vyhodnocování. Aplikace bude následně implementována pomocí vhodných nástrojů a technologií. V úvodu práce je pospáno co to vlastně webová aplikace a nástroje pro online průzkumy jsou. Dále je pospána specifikace a analýza aplikace, tedy požadavky na funkce, které by měla tato aplikace mít, včetně nástinu jejího uživatelského rozhraní. Následně je zde samotný návrh aplikace, ve kterém jsou popsány samotná data, tedy jak vypadá databáze spolu se vkládanými daty. V práci jsou také popsány použité technologie, včetně jejich základních popisů, výhod a nevýhod. V závěru jsou zhodnoceny dosažené cíle práce a také návrhy dalších možností, jak by se tato práce dala rozšířit.

Klíčová slova

Webová aplikace, Dotazník, Formulář, Sběr dat, Framework, Java, J2EE, Apache Wicket, Databáze, MVC

Abstract

The main objective of this bachelor is to design a web application for online surveys with statistical evaluation support. Applications will be implemented by using appropriate tools and technologies. In the introduction, it describes what actually web applications and tools for online surveys are. Furthermore, it will be describes the specification and analysis of applications, that implies the requirements for functions that should have this application, including the design of the user interface. Then there is the design of the application itself, which describes the data itself, which is how it looks this data in the database with inline data. In this documentation is also describes the technology which was used, including their basic characteristics, pros and cons. In conclusion are assessed achievements of the work and suggestions for future expansion of this application.

Key words

Web application, Form, Survey, Data collection, Framework, Java, J2EE, Apache Wicket, Database, MVC

Seznam použitých zkratek

Zkratka	Anglický význam	Český význam
OOP	Object-Oriented Programming	Objektově Orientované Programování
API	Application Programming Interface	Aplikační Programovací Rozhraní
http	Hypertext Transfer Protocol	Protokol pro přenos hypertextových dokumentů
i18n	Internationalization	Internacionalizace
W3C	World Wide Web Consortium	Konsorcium celosvětové sítě
JDBC	Java Database Connectivity	Rozhraní pro přístup k databázím v jazyce Java
API	Application programming interface	Aplikační rozhraní pro programování
SŘBD/DBMS	Database management system	Systém řízení báze dat
MD5	Message-Digest Algorithm	

Obsah

1 Úvod

V posledních letech zažívají webové aplikace obrovský rozvoj. Už to nejsou jen statické stránky, které slouží jako prezentace firem a organizací, ale začínají poskytovat i funkcionalitu. Funkcionalitu, která se dá použít pro nasměrování dalšího vývoje organizace či firmy na základě podnětů od uživatelů. K tomuto se dá využít buďto odkaz na firemní e-mail s tím, že takovéto podněty budou ručně zpracovány, nebo webové formuláře, které sběr a analýzu takovýchto dat zjednoduší a zrychlí. Tyto formulář by měli být dynamicky vytvořené, aby takovéto průzkumy mohli zadávat i lidé bez znalosti webových technologií.

Cílem této práce je navrhnout a implementovat takovouto webovou aplikaci. Pro implementaci tohoto systému bylo použito prostředí Java EE za použití webového frameworku.

Pro pochopení této práce, se čtenářům budou hodit alespoň elementární znalosti značkovacího jazyka HTML, programovacího jazyka Java, případně znalosti OOP.

2 Specifikace a analýza aplikace

Tato kapitola se zabývá specifikací a analýzou nástrojů pro online průzkumy s podporou statistického vyhodnocování. Na úvod této kapitoly, se seznámíme s tím, co to vlastně webová aplikace je a co představuje. Následně bude představena vize systému, kde se čtenář seznámí se základními problémy spojené s tvorbou nástrojů pro průzkumy. Dále budou popsány již existující nástroje podobného charakteru, jež jsou spuštěny na internetu, a které jsou v době psaní tohoto textu stále funkční. V posledních částech této kapitoly, budou pospány požadavky na systém, které byli získány analýzou již existujících nástrojů tohoto typu na internetu.

2.1 Webová aplikace

Webová aplikace v softwarovém inženýrství je aplikace poskytovaná uživatelům z webového serveru přes počítačovou síť Internet, nebo její vnitropodnikovou obdobu (intranet). Webové aplikace jsou populární především pro všudypřítomnost webového prohlížeče jako klienta. Ten se pak nazývá tenkým klientem, neboť sám o sobě logiku aplikace nezná. Jedním z důvodů pro vznik tohoto typu aplikací byla snaha o poskytnutí služeb softwarových aplikací bez složité a nákladné instalace na počítač klienta. Díky tomu stačí, aby měl klient na svém počítači nainstalovaný pouze internetový prohlížeč s příslušnými zásuvnými moduly. Použití tohoto typu aplikací oproti ostatním má obrovské výhody, uživatel se nemusí starat o specializovaný software, ani o aktualizace, neboť nejnovější verze webové aplikace je umístěna na přímo serveru. Zatímco vývojář/správce se stará pouze o jednu verzi aplikace, přičemž se nemusí starat o zpětnou kompatibilitu se staršími verzemi aplikace (s výjimkou konzistence dat v databázi). S centralizací aplikace souvisí i bezpečnost citlivých uložených dat, které stačí chránit na jednom místě.[1]

Zřejmou nevýhodou tohoto přístupu je vysoká závislost na poskytovateli aplikace a dostatečně dimenzované kapacitě připojení k serveru poskytovatele. Pokud se poskytovatel rozhodne ukončit poskytování této služby nebo ji přeruší z jiného důvodu, nelze službu nadále používat, na rozdíl od lokálně provozovaného software. Stejně tak pokud dojde k přerušení spojení se serverem poskytovatele, může být služba dočasně nedostupná. Tyto nevýhody jsou však vyváženy výhodou, kterou je na straně zákazníka prakticky nulová. Nebezpečnou hrozbou je přihlašování z veřejných počítačů (např. internetové kavárny), kde je vysoké riziko zneužití přihlašovacích údajů.

Při vývoji dnešních aplikací se klade důraz na asynchronní komunikaci se serverem a tím odstranění nevýhod při znovu načtení celé stránky, při jakékoliv interakci klienta se serverem. Pro vývoj takovýchto aplikací existuje celá paleta technologií. Pro zobrazení obsahu stránky a grafické úpravy ve webovém prohlížeči se nejčastěji používá značkovací jazyk HTML a kaskádové styly. Na straně serveru, který zpracovává klientské požadavky, je potřebný webový kontejner, který je zpracovává. Nejrozšířenějšími kontejnery jsou Apache Tomcat, GlassFish, JBoss a jiné. Pro implementaci aplikační logiky webové aplikace je možné si vybrat z velkého množství programovacích jazyků jako je Java, PHP, ASP.NET a jiné. Popisu použitých technologií a nástrojů je věnována část čtvrté kapitoly.

2.2 Vize systému

Dotazníky slouží ke zjišťování informací v populaci, nebo v nějaké menší skupině osob. Na jejich základě dochází k vyhodnocování určitých skutečností (ať už názorů, postojů či preferencí) a

orientací následných kroků. Dotazníky mohou být využity např. pro zjištění situace na trhu, průzkum veřejného mínění a podobně.

Z výše uvedeného vyplývá, že je potřeba vytvořit aplikaci, která uživateli nabídne komplexní správu dotazníku od jeho tvorby, přes sběr dat až po jejich vyhodnocení. Systém by měl sloužit k evidování všech vytvořených dotazníků a nasbíraných dat včetně základních informací o respondentech. Přímým důsledkem používání této webové aplikace je větší systematizace sběru dat oproti papírovým dotazníkům.

V případě papírových dotazníků je nutné dopředu vybrat lokalitu, kde se sběr uskuteční, což není ideální, protože v době sběru dat, se nemusí v dané lokalitě vyskytovat požadovaný počet respondentů v rámci cílové skupiny. Naproti tomu v případě online dotazníků, (i díky velkému rozmachu sociálních sítí) stačí jen umístit dotazník na nějakou stránku a přes různé služby (od emailů, přes komunikaci na fórech až po reklamy) na něj nasměrovat uživatele.

Nevýhodou papírových dotazníků jsou náklady na zisk dat, relativně pomalé vyhodnocování nasbíraných dat (data se musí přepsat do počítače, nebo ručně přetřídit). Oproti tomu, největší nevýhodou online dotazníků je anonymita uživatele, protože není možné nijak ověřit, zda respondent pomocí různých metod (např.: hlasování na jiném počítači, použití anonymizéru aj.) hlasoval vícekrát. Tato nevýhoda se dá omezit vhodným umístěním odkazu (např. umístěním odkazu na tematických fórech), avšak nedá se zcela vyřešit. Protože jakékoliv větší omezení (např. nutnost registrace, posílání SMS kódu aj.) by mohlo odradit případné respondenty a případné škodlivé respondenty, kteří chtějí statisticky ovlivnit výsledky dotazníku, to stejně nezastaví. Proto je nutné najít způsob jak co nejvíce omezit restrikce pro běžné respondenty a pro ostatní co nejvíce ztížit.

2.3 Existující nástroje na internetu

Na internetu existuje velká škála nástrojů pro online průzkumy a všechny nelze v rozumném čase prozkoumat a popsat. Z toho důvodu byl vybrán jen malý reprezentativní vzorek jak zahraničních, tak i tuzemských webových aplikací pro vytváření a správu dotazníků na internetu. Při výběru bylo přihlédnuto k pozici takovýchto aplikací v internetových vyhledávačích a k hodnocení na některých zpravodajských serverech.

2.3.1 Kwiqpoll

Kwiqpoll je nástroj, který zjednoduší Vaši práci při tvorbě anket. Tento nástroj je zcela zdarma a umožní Vám vytvořit jednoduché ankety v anonymním režimu (tj. bez nutnosti registrace). Nevýhodou u této aplikace je, že anketa je omezena na jedinou otázku a maximálně pět odpovědí. Ve statistickém vyhodnocování ankety je samozřejmě uveden počet respondentů, jejich hlasy a stát, ze kterého respondenti hlasovali.

2.3.2 Survey Monkey

Survey Monkey je v základní verzi bezplatný nástroj, který slouží k provádění online průzkumů. Lze v něm pohodlně vytvořit dotazník, rozeslat jej respondentům a zkontrolovat nasbíraná data. Životní cyklus dotazníku v Survey Monkey je následující: nejdříve vytvoříte dotazník, poté sbíráte odpovědi a pak analyzujete výsledky. Při tvorbě dotazníku si můžete vybrat z velkého množství

předpřipravených obsahových šablon (některé z nich jsou však přístupné pouze v placené verzi nástroje). V šablonách najdete například témata, která se zabývají marketingem, vzděláváním, politikou a podobně. V dalším kroku si můžete vybrat grafickou podobu dotazníku, přičemž na výběr máte opět z několika vizuálních šablon. Pak už zbývá jen přidat samotné otázky, obrázky nebo texty. Na konci je vám vygenerována URL adresa, kterou můžete sdílet se svými respondenty. Svůj dotazník také můžete zpětně upravovat a také ho podrobněji nastavit. Například možnost odpovídat pouze jednou z jednoho počítače, umožnění editace odpovědí, zobrazování výsledků nebo ukládání respondentovy IP adresy. Survey Monkey zároveň odpovědi analyzuje. Jednotlivé odpovědi si můžete procházet. Nevýhodou této aplikace je, že neumožňuje vytvoření anonymních dotazníků a většina funkcí spojená s analýzou a vyhodnocováním dotazníků je placená.

2.3.3 MojeAnketa

Nyní se dostáváme k zástupci tuzemských nástrojů pro tvorbu dotazníků, tím je nástroj MojeAnketa. Tento nástroj je zcela zdarma, ale pro vytváření vlastních dotazníků, je nutná registrace. Tvorba dotazníku je v zásadě stejná jako u Survey Monkey. Výsledky jsou zobrazeny přehledně. Jednotlivé výsledky odpovědí na otázky si může uživatel prohlédnout v několika různých grafech (konktrétně: pruhový, sloupcový, výsečový a spojnicový). Uživatel má také možnost prohlédnout si jednotlivé odpovědi respondentů a také výsledky exportovat do XLS formátu.

2.4 Funkční požadavky

Ve vizi jsme specifikovali základní problémy a požadavky webové aplikace, které následně byli rozšířeny analýzou již existujících nástrojů na internetu. Nyní je potřeba definovat si požadavky na funkčnost aplikace detailněji. V aplikaci je nutné, propojit konkrétní dotazník s konkrétním uživatelem, který ho vytvořil. Tím se zabrání nechtěným úpravám, ze strany třetí osoby. Pro toto, je nejvhodnější použít systém registrace a autentizace. Každý registrovaný uživatel bude mít možnost vytvoření, editace a smazání vlastního dotazníku. Dotazník se může skládat s více různých menších dotazníků, a proto je vhodné, aby jednotlivé komponenty byli standardizované.

Dále je žádoucí, aby na dotazník odpovídalo co nejvíce respondentů, což se dá docílit vytvořením speciálního odkazu, které uživatel zkopíruje a umístí jej na vhodné místo, nebo kusu kódu, který může být vložen do přímo do HTML kódu stránek zadavatele. Pro co nejpřesnější sběr dat, je žádoucí, aby každý respondent na daný dotazník odpovídal pouze jednou.

Vzhledem k obecnosti (tato aplikace není navrhována pro konkrétního odběratele, který by měl předem stanoveny požadavky) této webové aplikace, není žádoucí, aby se případný respondent musel registrovat v této aplikaci, protože by ho to mohlo odradit od případného hlasování. Proto bude nejvhodnější pro zabezpečení tohoto použít ukládání IP adresy respondenta a jako redundanci ukládání informace o daném dotazníku do cookies. Dále je nutné po hlasování respondentovi oznámit, zda byl jeho hlas uložen a případně zobrazit graf s odpověďmi (v závislosti na typu dotazníku). Pro tazatele je třeba vytvořit seznam s odpověďmi jednotlivých respondentů na danou anketu a k tomuto seznamu vytvořit grafy a statistiky, které přehledně zobrazí nejčastější odpovědi na jednotlivé otázky. Nakonec, vzhledem ke globalizaci je vhodné použít nástroje pro internacionalizaci a lokalizaci stránek, tak aby aplikaci mohli používat i uživatelé, kteří nehovoří česky.

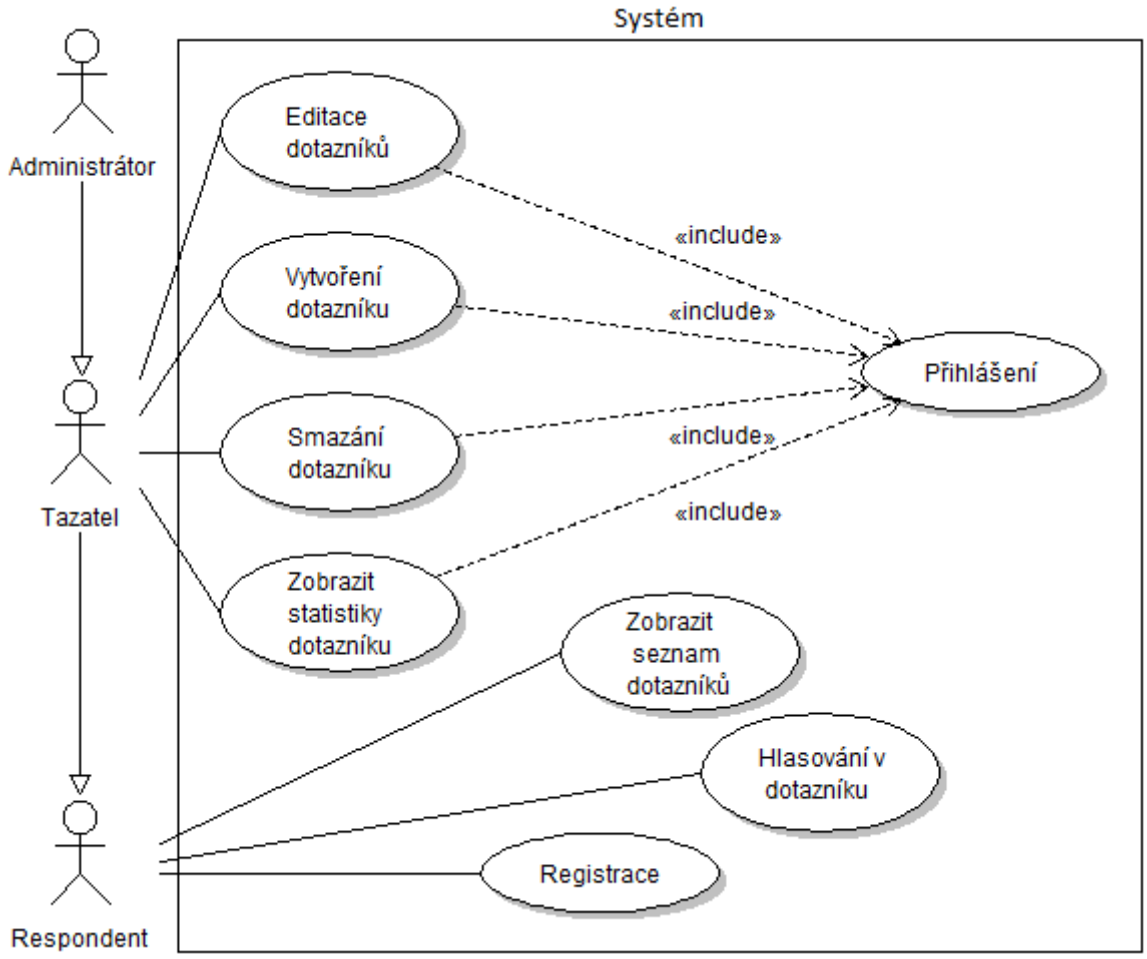
2.5 Diagram případů užití

Diagram případů užití (Use Case Diagram) zachycuje vnější pohled na modelovaný systém a tím pomáhá odhalit hranice systému a slouží jako podklad pro odhady rozsahu. Jde o posloupnost souvisejících transakcí mezi účastníkem (zpravidla uživatelem v určité roli, ale také jiným systémem) a systémem během vzájemného dialogu. Hlavním účelem je zachycení aktérů, kteří se systémem komunikují a vztahů mezi službami a těmi, kterým jsou poskytovány, a to vizuální i textovou podobou, která je srozumitelná vývojářům systému i zákazníkům (tj. těm, kteří jej mají používat). Na obrázku 3.1je názorně ukázaný případ užití. Případy užití byly získány z analýzy funkčních požadavků a přiřazeny k aktérům. Dále v textu následuje popis aktérů.

Administrátor – Administrátor má stejná práva jako tazatel či respondent, navíc má přístup ke zdrojovým kódům aplikace a přímý přístup do databáze, takže může měnit obsah stránek ručně. V rámci aplikace však není nijak rozlišen od tazatele.

Tazatel – Tazatel má v systému důležitou roli, vytváří nové dotazníky. Dále má právo editovat, mazat dotazníky a prohlížet si statistiky dotazníků

Respondent – Respondent představuje aktéra, který má právo hlasovat ve zveřejněných dotaznících



Obrázek 2.1: Diagram případů užití

Dále následuje popis jednotlivých případů užití prostřednictvím mini-specifikací. Vzhledem k počtu případů užití uvádím pouze některé z nich. Zbylé mini-specifikace budou uvedeny v příloze.

Tabulka.1: Případy užití: Vytvoření dotazníku

Případ užití	Vytvoření dotazníku
Aktér	Tazatel
Vstupní podmínky	1. Aktér je přihlášen do aplikace
Výstupní podmínky	
Záruky úspěchu	1. Dotazník je uložen do databáze
Minimální záruky	1. Aplikace upozorní, že dotazník nebyl uložen
Hlavní scénář	1. Případ užití začíná, když aktér vybere “Vytvoření dotazníku“ 2. Aplikace zobrazí formulář pro tvorbu dotazníku 3. Aktér vyplní požadované informace a klikne pokračovat ve tvorbě dotazníku

	4. Systém zkontroluje vyplněné údaje 5. KDYŽ údaje jsou správné 5.1 Aplikace zobrazí formulář pro zadání otázky a odpovědí 5.2 Aktér vyplní požadované informace 5.3 Systém zkontroluje vyplněné údaje 5.4 KDYŽ klikne **dokončit** dotazník, celý dotazník je uložen do databáze 5.5 KDYŽ klikne **pokračovat**, pokračuje se krokem 5.1 6. JINAK aplikace vypíše chybové hlášení a pokračuje krokem 2.

Tazatel zvolí vytvoření dotazníku. Po vyplnění požadovaných údajů vybere možnost Pokračovat v tvorbě dotazníku a následně je zkontrolována správnost zadaných údajů. Při špatném vyplnění některých údajů systém vypíše chybové hlášení, čímž upozorní uživatele, kde nastala chyba a zároveň mu nabízí chybné údaje opravit. Pokud jsou zadané údaje správné, systém zobrazí další formulář, tentokrát pro zadání otázek a odpovědí na daný dotazník. Po zadání požadovaných údajů systém zkontroluje vyplněné údaje, pokud jsou některé chybně zadané, vypíše chybové hlášení a nabídne aktérovi chybné údaje opravit. Pokud aktér klikne **dokončit** dotazník, celý dotazník je uložen do databáze a uživatel je přesměrován na detail dotazníku. Pokud aktér klikne **pokračovat**, zobrazí se nový formulář pro zadání otázek a odpovědí pro stejný dotazník.

Tabulka.2: Případy užití: Hlasování v dotazníku

Případ užití	Hlasování v dotazníku
Aktér	Respondent, Tazatel
Vstupní podmínky	2. Dotazník je aktivní
Výstupní podmínky	
Záruky úspěchu	1. Odpovědi na daný dotazník jsou uloženy do databáze a zpřístupněny pro tazatele
Minimální záruky	1. Aplikace upozorní, že na dotazník již není možné odpovídat
Hlavní scénář	1. Případ užití začíná, když aktér zvolí dotazník, na který chce odpovídat 2. Aplikace zobrazí dotazník 3. Aktér vyplní požadované údaje 4. Systém údaje zkontroluje 5. KDYŽ jsou všechny požadované údaje vyplněny 5.1 Data jsou uloženy a aplikace zobrazí statistiky

	6. JINAK systém vypíše chybové hlášení a pokračuje se krokem 3.

Aktér, vyplní požadované údaje dotazníku, které systém následně zkontroluje. Pokud jsou všechny údaje zadány správně, potom odpovědi z dané instance dotazníku jsou uloženy do databáze. Pokud jsou zadaná data neúplná či chybná, pak aplikace zobrazí chybové hlášení a umožní aktérovi opravit požadované údaje.

2.6 Diagram aktivit

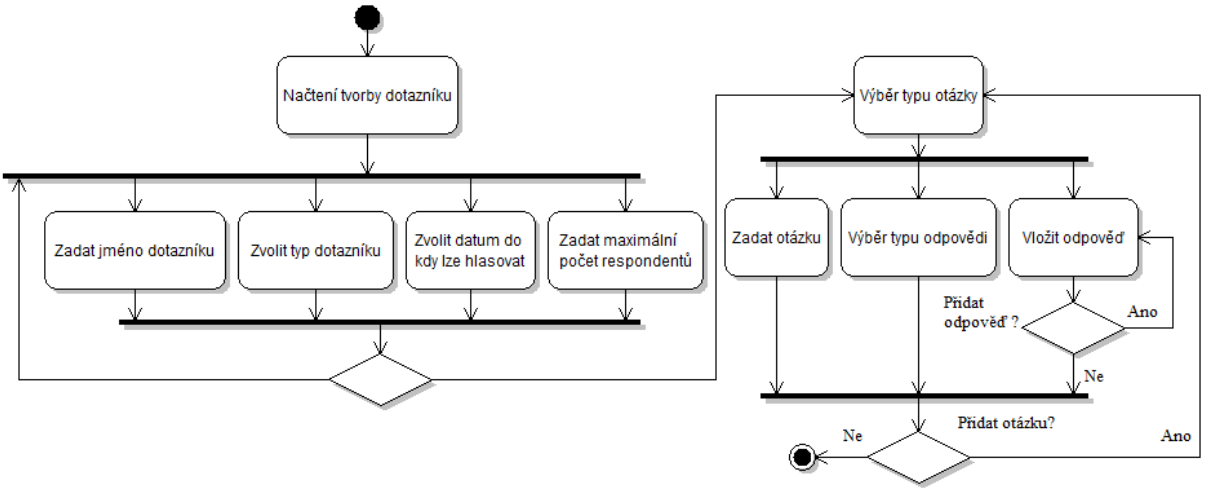
Diagramy aktivit jsou vývojové diagramy, které umožňují modelování procesu jako aktivity, kterou tvoří uzly a jejich spojení (hrany). V tomto případě jsou připojeny k (vybraným) případům užití a snaží se tak graficky znázornit jejich scénáře.

V rámci analýzy této webové aplikace bylo rozpoznáno několik procesů, které v podstatě odpovídají případům užití. Z nich pak byly vybrány ty s netriviálním průběhem. Jejich diagramy aktivit dle specifikace UML2 a popis jsou uvedeny v následujícím textu

Tvorba dotazníku

Vytvoření dotazníku je důležitý proces, protože nelze sbírat bez toho, aby dotazník byl vytvořen a uložen v databázi. Diagram k této aktivitě je znázorněn na obrázku 2.2.

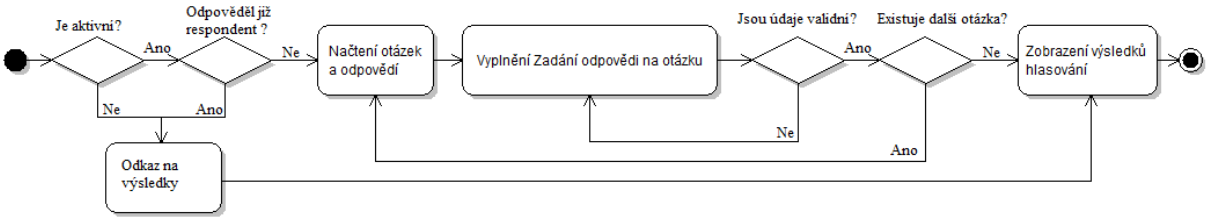
První akcí je načtení tvorby dotazníku. Jakmile je tato možnost načtena, objeví se formulář, který je nutno vyplnit, nezáleží na pořadí vyplnění. Pro úspěšné pokračování je nutné vyplnit jméno dotazníku a zvolit typ dotazníku. Typ dotazníku určuje, o jaký typ dotazníku se jedná (na výběr jsou: anketa, sběr dat a testy). Zvolení data do kdy lze hlasovat ani maximální počet respondentů není vyžadován, avšak při zadání musí být vyplněny korektně. Následně systém provede validaci zadaných údajů, v případě chyby vypíše chybové hlášení a nabídne možnost opravit zadané údaje. Následuje výběr typu otázky (buďto jednoduchá, nebo maticová). Po této volbě je nutné zadat otázku/y, vybrat typ odpovědi (odkazy, přepínací, zaškrtávací pole nebo textové pole). Posléze je nutno zadat odpověď na otázku, následně je možno přidat další odpověď a vyplnit ji. Nakonec v případě výběru přidat otázku, se pokračuje znovu od aktivity výběr typu dotazu. Jinak je dotazník uložen do databáze a připraven k publikování.



Obrázek 2.2: Diagram aktivit pro tvorbu dotazníku

Hlasování

Ač tento proces je triviální, tak i přesto se jedná o hlavní proces této webové aplikace. Nejprve se zkontroluje, zda je dotazník stále aktivní (zda již nedosáhl maximálního počtu respondentů, není po datu, do kdy lze hlasovat, či zadavatel neukončil sběr dat). Pokud již nadále není dotazník aktivní, systém respondentovi znemožní odpovědět a nabídne mu odkaz, přes který se může podívat na výsledky. Pokud dotazník je stále aktivní pokračuje kontrolou, zda respondent na daný dotazník již odpověděl, pokud ano, aplikace mu znemožní odpovědět a nabídne mu odkaz, který jej přesměruje na výsledky. Jinak respondent pokračuje aktivitou načtení otázky. Po načtení Otázky (Více otázek v případě maticového typu sub-dotazníku) a odpovědí, uživatel vyplní či vybere odpověď/di. V případě nějaké nekorektnosti, aplikace zobrazí chybové hlášení, o chybějící odpovědi



Obrázek 2.3: Diagram aktivit pro hlasování

2.7 Uživatelské rozhraní

Uživatelskému rozhraní nebyla dosud věnována větší pozornost, avšak částečně navazuje na předchozí diagramy z této kapitoly, které popisují funkční strukturu. Protože se jedná o webovou aplikaci, přístup bude probíhat pomocí webového prohlížeče, který nabídne webové rozhraní.

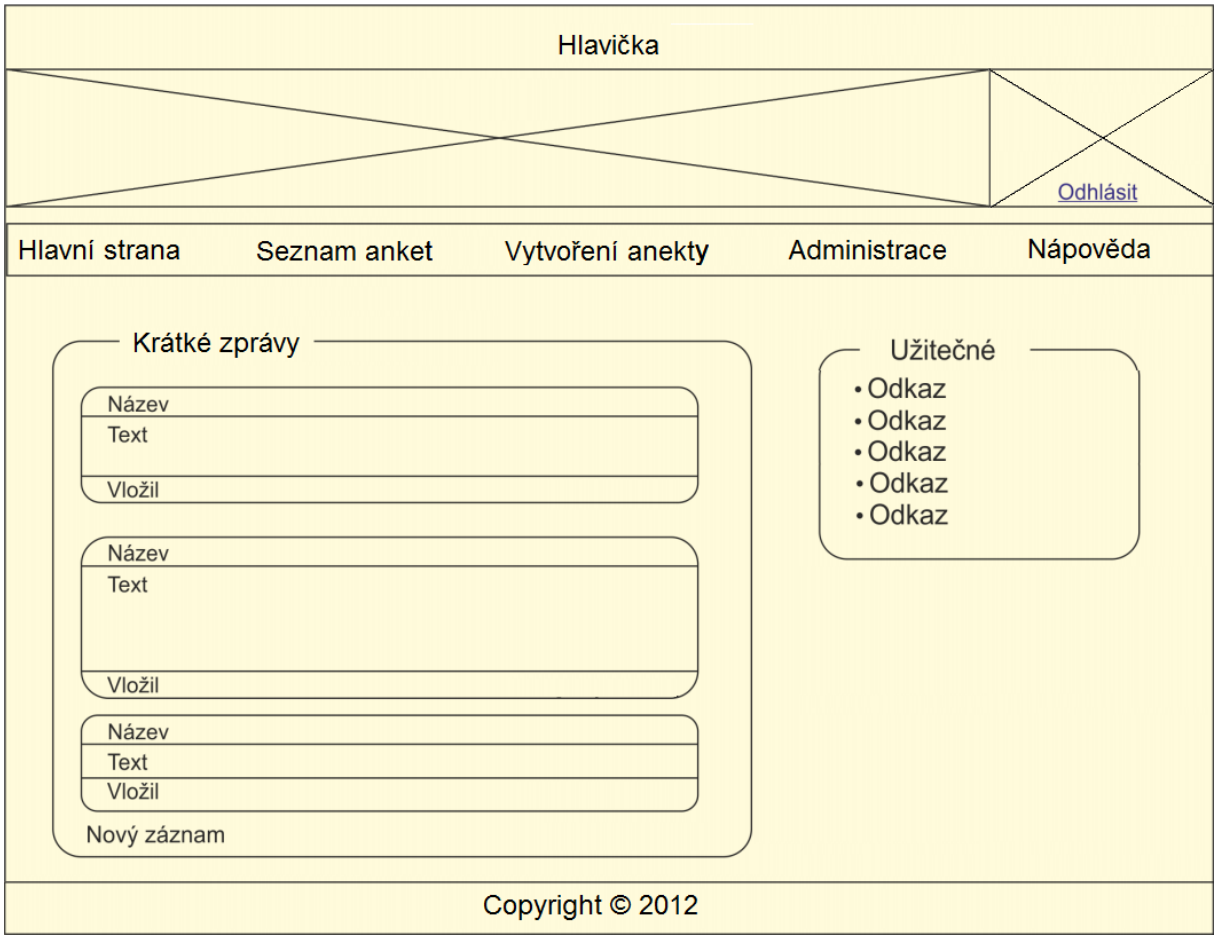
2.7.1 Společné části stránky

Webové rozhraní, bude obsahovat společnou část pro většinu (Pro stránku zobrazující dotazník umístěný na nějaké jiné stránce, není žádoucí, aby se tam zobrazovali věci jako menu a jiné.) podstránek aplikace. Tuto část bude tvořit hlavička s logem stránky, v jejímž pravém horním rohu

bude v rámci hlavičky zobrazen panel pro přihlášení a registraci, nebo odhlášení uživatele, v levém horním rohu bude zobrazena možnost přepnutí lokalizace. Dále pod logem stránky (stále v rámci hlavičky) bude zobrazeno horizontální menu, kde budou zobrazeny odkazy, zpřístupňující jednotlivé funkce aplikace. Budou existovat dvě verze menu, jedna verze pro přihlášeného uživatele a druhá pro nepřihlášeného uživatele.

2.7.2 Hlavní stránka

Po načtení webové aplikace, přivítá uživatele hlavní stránka, které dominuje seznam krátkých zpráv. Tyto krátké zprávy jsou viditelné pro přihlášené i nepřihlášené uživatele. Vkládat je však budou moci pouze přihlášení uživatele. Po pravé straně bude mít uživatel seznam užitečných odkazů (např.: odkazy na stránky, kde lze umístit dotazníky a jiné), jejichž seznam určuje administrátor. Toto jest znázorněno na obrázku 2.4.

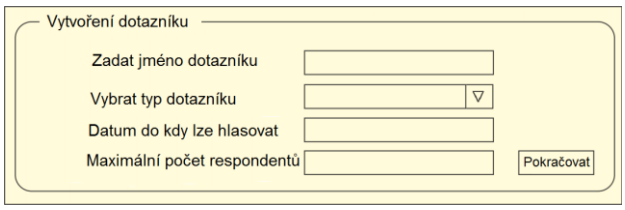


Obrázek 2.4: Návrh hlavní stránky

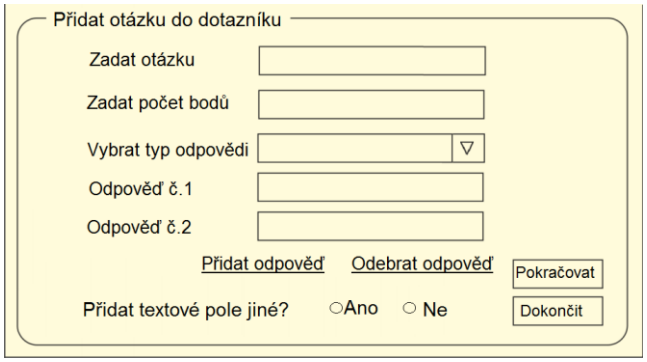
2.7.3 Stránka pro vytvoření dotazníku

Tato stránka bude obsahovat několik panelů (panel je znovu použitelná komponenta, která může obsahovat několik komponent a která může obsahovat stejné prvky jako stránka, avšak nemůže být zobrazena samostatně). V první panelu bude zobrazen formulář pro vložení jména

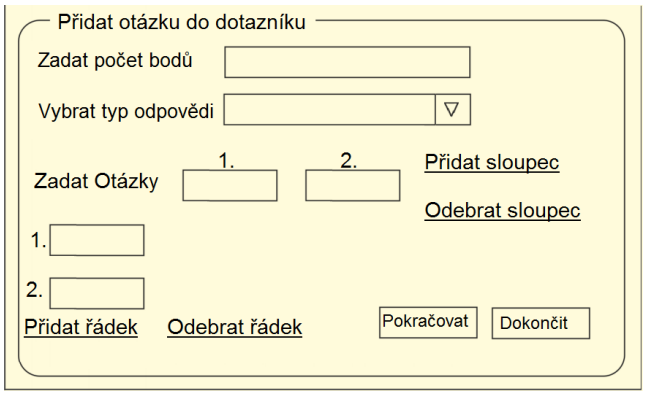
dotazníku, výběru typu dotazníku, zadání data platnosti dotazníku a zadání maximálního počtu respondentů. Po té je zobrazeno výběrové pole s možností výběru jednoduchého sub-dotazníku (viz obrázek 2.6) nebo maticového sub-dotazníku (obrázek 2.7). U obou těchto dotazníků je kolonka *zadat počet bodů.* Tato kolonka se vyskytuje pouze u dotazníků, které jsou označeny jako testovací. Dále mají oba tyto sub-dotazníky společnou kolonku typ odpovědi (rozsáhlejší popis tohoto v podkapitole 3.5.5 Standardizované komponenty). V případě jednoduchého sub-dotazníku, tazatel, kromě výše zmíněného zadá otázku a poté odpověď/di. Pomocí Odkazu *Přidat odpověď*, je tazateli zobrazeno nové textové pole pro další odpověď. Pomocí odkazu *Odebrat odpověď*, je tazateli odebráno poslední textové pole, přičemž minimální počet odpovědí jsou dvě. V případě maticového sub-dotazníku je možno zadat více otázek. Tyto otázky jsou buďto ve sloupcích, nebo v řádcích.



Obrázek 2.5: Panel pro tvorbu dotazníků



Obrázek 2.6: Panel pro přidání jednoduché otázky



Obrázek 2.7: Panel pro maticový sub-dotazník

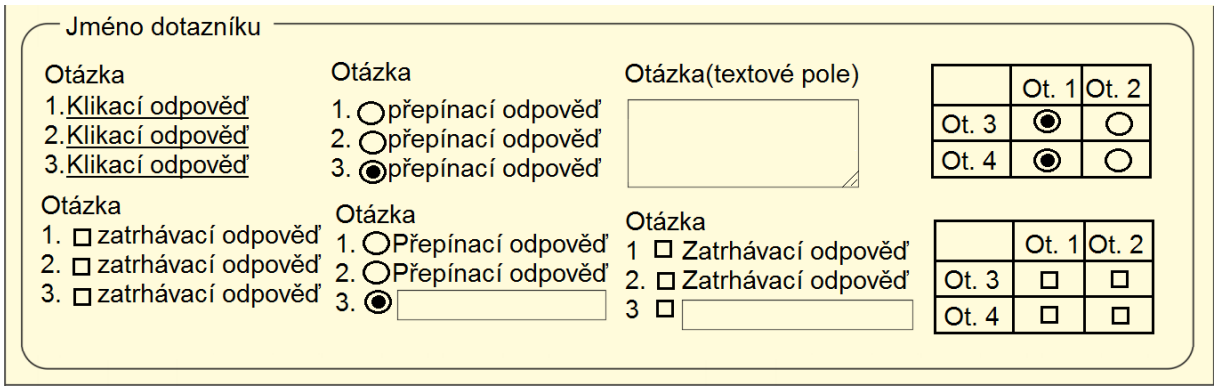
2.7.4 Seznam dotazníků

Seznam dotazníků bude obsahovat tabulku se sloupci identifikační řetězec, jméno, autor, typ, datum vytvoření, datum platnosti a počet respondentů dotazníku. V této tabulce dále budou čtyři slupce s funkčností hlasovat, detail, úprava a smazat dotazník. Funkce hlasovat bude přístupná pro

všechny uživatele, avšak funkce detail dotazníku, úprava dotazníku a smazat dotazník, bude přístupná jen přihlášenému autorovi daného dotazníku. Přes detail, bude mít uživatel přístup k úpravě a statistikám dotazníků.

2.7.5 Standardizované komponenty pro dotazníky

Pro vykreslení dotazníku je nutné standardizovat vzhled vykreslovaných otázek a odpovědí. Jednotlivé komponenty jednoduchého sub-dotazníku mají společné zobrazení otázky nad seznamem odpovědí. Klikací odpověď je tvořená odkazem, respektive seznamem odkazů, s hodnotou odpovědí vztahující se k dané otázce. Tento typ odpovědí lze vybrat pouze při výběru typu dotazníku *anketa*. Seznam přepínacích odpovědí je dostupný při jakémkoliv typu dotazníku, je představován komponentou typu *radio button*, tato komponenta dovoluje vybrat pouze jednu odpověď. Dále speciální případ seznamu přepínacích odpovědí, s textovým polem pro vlastní odpověď. Toto je dostupné pro sub-dotazník, u kterého bylo vybráno *přidat textové pole jiné*. Seznam zatrhávacích odpovědí je tvořen komponentou *checkbox*, která dovoluje vybrat více odpovědí. U tohoto seznamu odpovědí je stejně jako u seznamu přepínacích odpovědí, možnost přidat textové pole pro vlastní odpověď. Další komponentou jednoduchého sub-dotazníku, je otázka se seznamem odpovědí, u kterých je možno vepsat ohodnocení(například důležitost) odpovědi. Poslední komponentou jednoduchého sub-dotazníku je textové pole. Maticový sub-dotazník je tvořen seznamem otázek umístěných v hlavičce tabulky (tj. první řádek a první sloupec tabulky). V rámci tvorby maticového sub-dotazníku lze vybrat mezi přepínacími odpověďmi a zaškrtávacími odpověďmi. V případě přepínacích odpovědí lze vybrat jednu odpověď pro každý řádek. Jednotlivé komponenty jsou k vidění na obrázku 2.8.



Obrázek 2.8: Standardizované komponenty

3 Návrh aplikace

Tato kapitola navazuje na předchozí specifikaci a analýzu a zaobírá se návrhem informačního systému.

3.1 Zjednodušený ER diagram databáze

ER diagram (Entity-Relationship diagram) je speciální typ grafu, který znázorňuje vztahy mezi entitami (Entita – jednoznačně identifikovatelný objekt reálného světa v tomto případě záznam v tabulce).

Users (Uživatelé) – Tabulka slouží pro uložení informací o uživatelích

ShortMessage(Krátké zprávy) – Slouží po uchovávání a zobrazení zpráv uživatelů

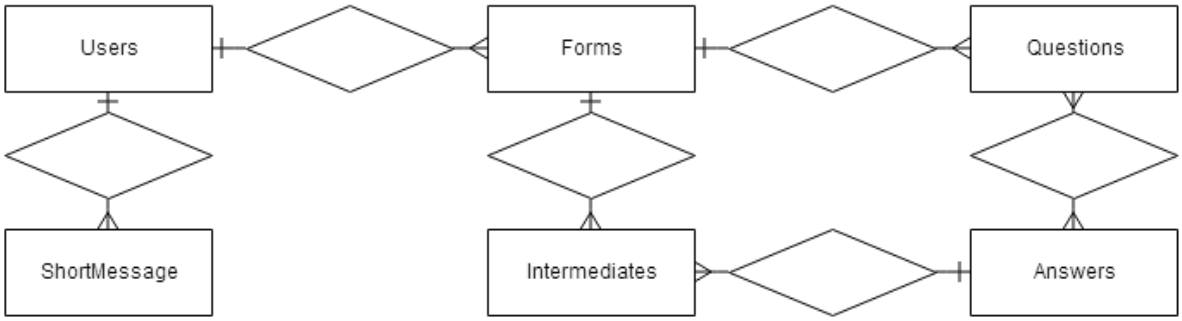
Forms(Dotazníky) – Slouží pro uložení základních informací o dotaznících. Obsahuje jeden cizí klíč z tabulky *Users*, který určuje, kdo je tvůrcem dotazníku.

Questions(Otázky) – Obsahuje otázky, které jsou propojené s tabulkou *Forms*

Answers(Odpovědi) – Do této tabulky jsou ukládány odpovědi na otázky

QuestionAnswers(Otázky-Odpovědi) – Vazební tabulka pro tabulky otázek a odpovědí.

Intermediates(Výsledky) – Uchovává data s odpovědmi respondentů na dané dotazníky.



Obrázek 3.1: Zjednodušený ERD

Uživatelé, Krátké zprávy (Users, ShortMessage) 1:N

Uživatelé, Dotazníky (Users, Forms) 1:N

Dotazníky, Otázky (Forms, Questions) 1:N

Otázky, Odpovědi (Questions, Answers) M:N

Otázky, Výsledky (Questions, Intermediates) 1:N

Dotazník, Výsledky (Forms, Intermediates) 1:N

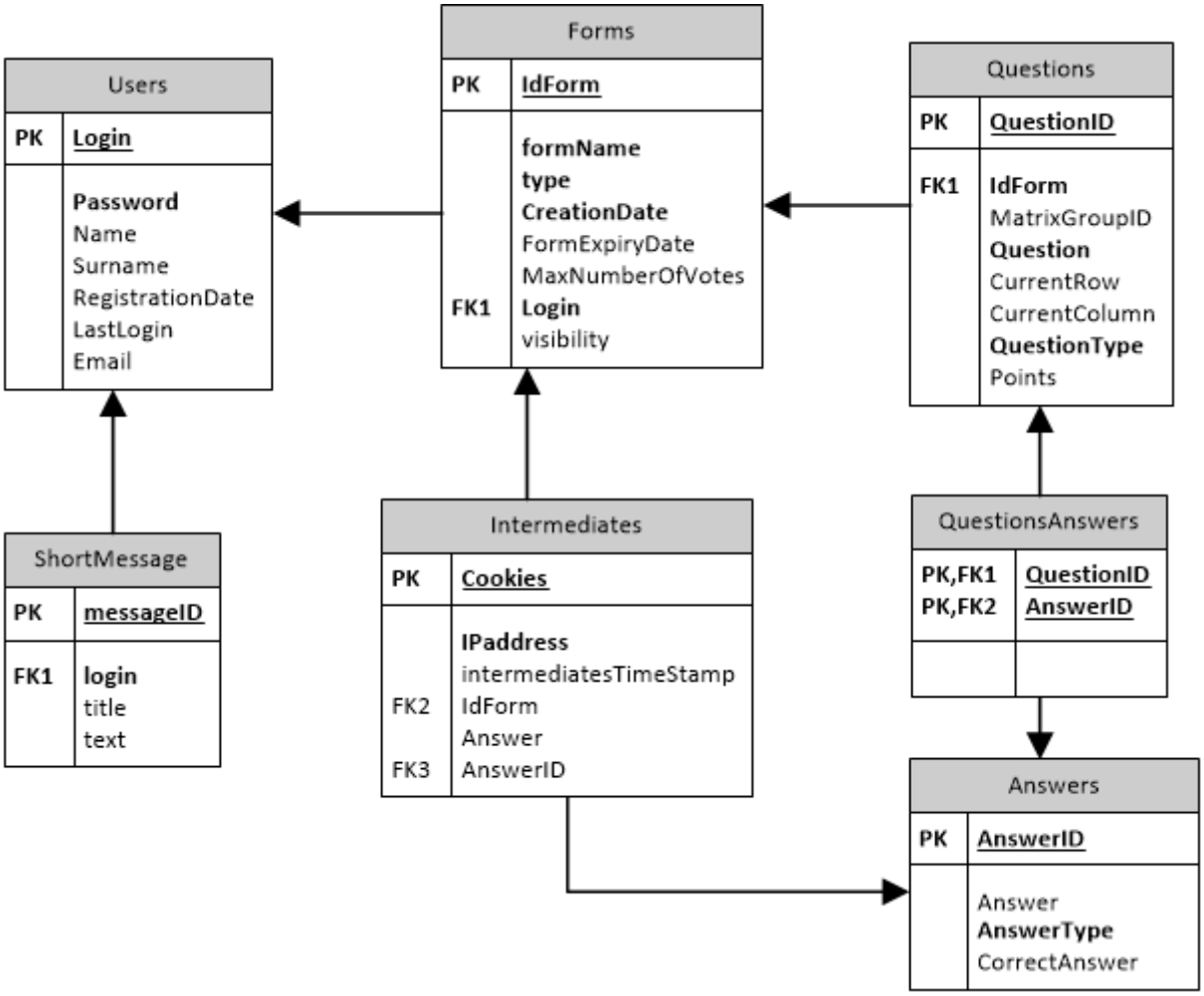
U vztahu mezi tabulkou Otázky (Questions) a Odpovědi (Answers) je kardinalita vztahu M:N, to znamená, že pro více otázek v databázi, existuje více odpovědí (ve webové aplikaci k tomuto stavu dojde, když je jako typ otázky vybrána maticový sub-dotazník). Relační databáze nejsou schopny takovýto stav adekvátně zaznamenat. Relační databáze pracuje nejlépe s kardinalitou, kdy jednomu záznamu z jedné tabulky odpovídá jeden či více záznamů z tabulky druhé (např. jeden uživatel vytvoří několik dotazníků). Pro takovýto rozklad se používají vazební tabulky, které ukládají primární klíče v tomto případ z tabulky *Otázky* a *Odpovědí* do nové tabulky. V případě této aplikace však dochází jen ke kardinalitě 2:N (tj. nikdy nenastane případ, kdy pro jednu odpověď existuje více jak dvě otázky). Z tohoto důvodu není nutné použít vazební tabulku, jen je nutné doplnit tabulku odpovědí o jeden sloupec, do nějž se budou ukládat primární klíče sloupcových otázek maticových sub-dotazníků.

Podrobnější ER diagram, včetně datového slovníku je k dispozici v příloze.

3.2 Konceptuální datový model

Konceptuální či sémantické modely jsou pokusem umožnit vytvoření popisu dat v databázi nezávisle na jejich uložení. Konceptuální model představuje formální popis modelované reality. Hlavními úkoly je nalezení entit, vtahů a atributů. Sémantické modely slouží obvykle k vytvoření schémat s následnou transformací na databázové schéma. Spojíme-li sémantickou a databázovou úroveň, dostáváme se k tzv. objektově-orientovaným SŘBD. Konceptuální model aplikace je vytvořen na základě reality, není ovlivněn budoucími prostředky řešení. Jeho výhodou oproti fyzickému modelu je fakt, že při změně konkrétní databáze zůstává neměnný a návrh tak nemusí začínat od začátku.

Konceptuální struktura aplikace vznikala především na základě analýzy z předcházejících dvou kapitol a zachycena je pomocí entit a jejich vztahů na obrázku 3.2. V předešlé podkapitole bylo řečeno, že vazební tabulka nebude použita, avšak pro nástin toho jak taková vazební tabulka vypadá, je zde uvedena.



Obrázek 3.2: Konceptuální datový model

3.3 Použité návrhové vzory

V softwarovém inženýrství, návrhový vzor (anglicky design pattern) představuje obecné řešení problému, které se využívá při návrhu počítačových programů. Návrhový vzor není knihovnou nebo částí zdrojového kódu, která by se dala přímo vložit do našeho programu. Jedná se o popis řešení problému nebo šablonu, která může být použita v různých situacích. Objektově orientované návrhové vzory typicky ukazují vztahy a interakce mezi třídami a objekty, aniž by určovaly implementaci konkrétní třídy. Algoritmy nejsou považovány za návrhové vzory, protože řeší konkrétní problémy a nikoliv problémy návrhu.[2]

Tabulka.3: Tři základní skupiny návrhových vzorů

Vytvářející vzory	Factory Method, Abstract Factory Method, Builder, Prototype, Singleton
Strukturální vzory	Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy.
Vzory Chování	Chain Of Responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template, Visitor.

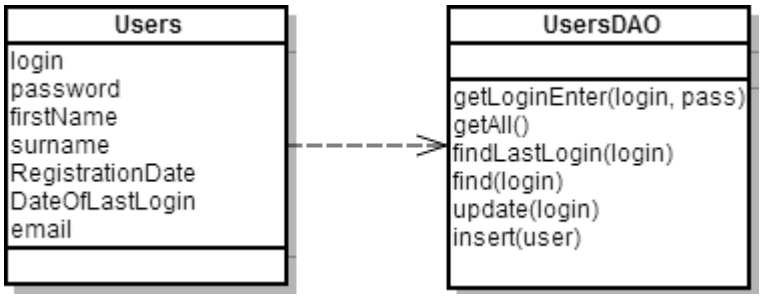
Data Access Object

Data Access Object (Přístup k datovým objektům) slouží jako přístup k databázové tabulce, komunikace s databází je obalena do jedné třídy, která data získává i čte, mapuje a vrací kolekci dat. Tento návrhový vzor byl vytvořen přímo pro prostřed Java EE.[4]

Tento návrhový vzor funguje jako brána, která zapouzdřuje databázové operace prováděné nad jednou databázovou tabulkou, bez toho aby odhalila detaily databáze. Každá databázová tabulka je v aplikaci reprezentována samostatnou třídou. Tato třída pak obstarává CRUD operace (tj. čtyři základní operace pro Vytvoření – Create, Čtení – Read, Editace – Update a Smazání – Delete záznamu v trvalém uložišti) s jednotlivými řádky tabulky. Metody, které v tabulce vyhledávají řádky ať už pomocí primárního klíče nebo jiných kritérií by měly vždy vracet množinu řádků a to i v případě, že výsledkem bude jen jeden řádek. Odvozenou třídu lze doplnit o metody, které např. zjednoduší vyhledávání nad konkrétní tabulkou podle specifického kritéria.

Tento návrhový vzor je jednou z metodik, jak zajistit ORM. ORM neboli objektově relační mapování je programovací technika, která zajišťuje automatickou konverzi dat mezi relační databází a objektově orientovaným jazykem. Nevýhodou tohoto návrhového vzoru je, že když je změněna tabulka v databázi, musí se pozměnit obě třídy.

V aplikaci má každá tabulka je reprezentována svým modelem, to je třída, která obsahuje proměnné stejného jména a typu jako databázová tabulka. Třídy pro přístup k datům mají vždy za svým jméno uvedenou zkratku DAO (Data Access Object). Tento návrhový vzor zajišťuje velice podobnou funkcionalitu jako vzor Table Data Gateway.[2]



Obrázek 3.3: Vzor data access object

Users – model, reprezentující jeden řádek dat databáze z tabulky uživatel (resp. Jednoho uživatele).

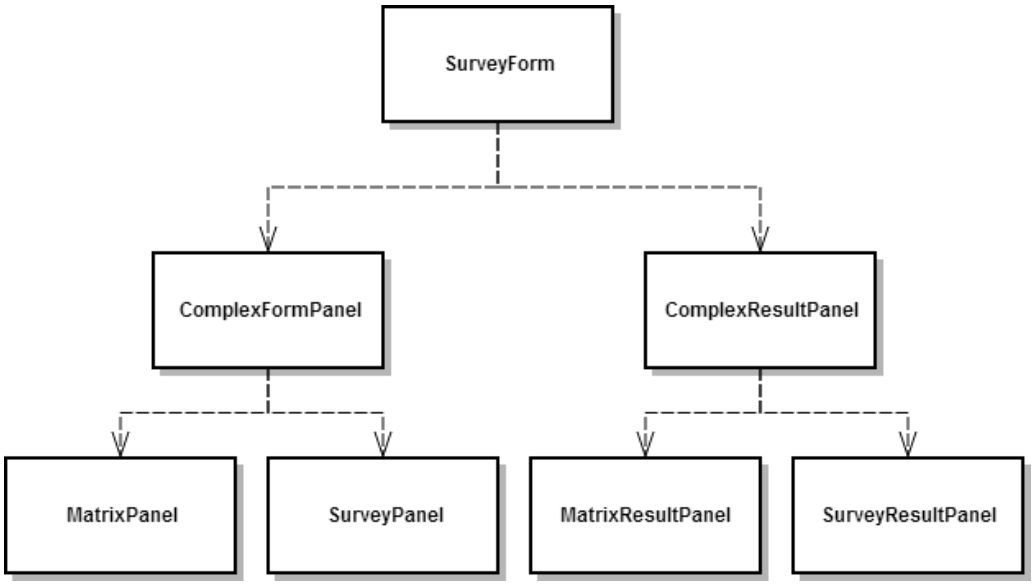
UsersDAO – umožňuje přístup k datům pro třídu Users

Návrhový vzor byl použit ve webové aplikaci pro modelování přístupu k datům pro všechny objekty problémové domény. K vidění na obrázku 3.3.

Composite

Návrhový vzor Composite (dále jako kompozit) je typickým představitelem strukturálních návrhových vzorů. S jeho pomocí je možné organizovat objekty do stromové struktury, přičemž ke každému jednotlivému objektu je možné přistupovat identickým způsobem. Mnohdy v naší webové aplikaci pracujeme s několika rozhraními různých tříd. To může být často nepřehledné a problematické, zejména pokud jsou třídy různě složité. Pokud spolu třídy logicky souvisí, můžeme

k nim přistupovat pomocí kompozitu. Získáme tak jednotné rozhraní pro funkcionalitu, kterou nám kompozit poskytuje. Kompozit je poměrně jednoduchý vzor, který se skládá z jedné třídy, která kompozit tvoří. Ta poskytuje rozhraní jak pro jednoduché tak i pro složité třídy. Celá složitá struktura tříd je v pozadí.



Obrázek 3.4: Návrhový vzor kompozit

Tento návrhový vzor byl v této webové aplikaci použit vícekrát. Na obrázku 4.2 je k vidění případ, kdy potřebujeme vykreslit celý dotazník. Třída SurveyForm reprezentuje stránku, na která má informaci o tom jaký dotazník má být vykreslen, ale celá logika, jíž se řídí vykreslování dotazníku, je naspána ve třídě ComplexFormPanel, jež rozhoduje, na základě načtených otázek, zda bude vykreslen Maticový sub-dotazník (MatrixPanel) nebo jednoduchý sub-dotazník (SurveyPanel), Tyto třídy již vykreslují celý sub-dotazník s otázky a odpověďmi. Třída ComplexResultPanel má podobnou funkci jako třída ComplexFormPanel, avšak vykresluje již výsledky dotazníku.

4 Implementace

Samotné vlastní implementaci předchází výběr z pestré škály moderních technologií, které lze použít k vývoji webových aplikací. Následuje popis zvolené architektury a základních postupů typických pro vybrané technologie. Pro demonstraci typických rysů zvolených nástrojů nechybí ukázky vybraných zdrojových kódů. Kapitola si neklade za cíl vysvětlit dopodrobna použitou platformu, ale zdůraznit její základní principy.

4.1 Webové frameworky pro Java EE

V práci často pracuji s pojmem framework (česky rámec). Český překlad není v odborných kruzích příliš využíván, proto nadále budu v textu pracovat s pojmem framework s použitím českého skloňování. Framework je softwarová struktura, která slouží jako podpora při programování, vývoji a organizaci jiných softwarových projektů. Frameworky mohou obsahovat podpůrné programy, API knihovny, implementace návrhových vzorů aj. Tyto pak programátorovi šetří čas a řádky kódu, protože programátor se nemusí zajímat o základní nastavení a psát stejný, či podobný kód dokola (např.: u frameworků pro práci s grafy vývojáři odpadá nutnost psát kód pro vytvoření grafu, stačí mu jen svázat či propojit data s instancí tříd z tohoto frameworku).

4.1.1 Funkcionalita webových frameworků

Úlohou webových frameworků je zjednodušit množství běžných a opakovaných aktivit při vývoji aplikací a poskytnout ověřené a funkční řešení pro nejčastější problémy jako jsou např.: validace formulářů, validace dat a podobně. Zde uvádím několik základních funkčních požadavků:

- Jednotný vzhled stránek
- i18n a lokalizace
- validace vstupních dat
- uchování informací o relaci uživatele
- generování webových stránek ze šablon

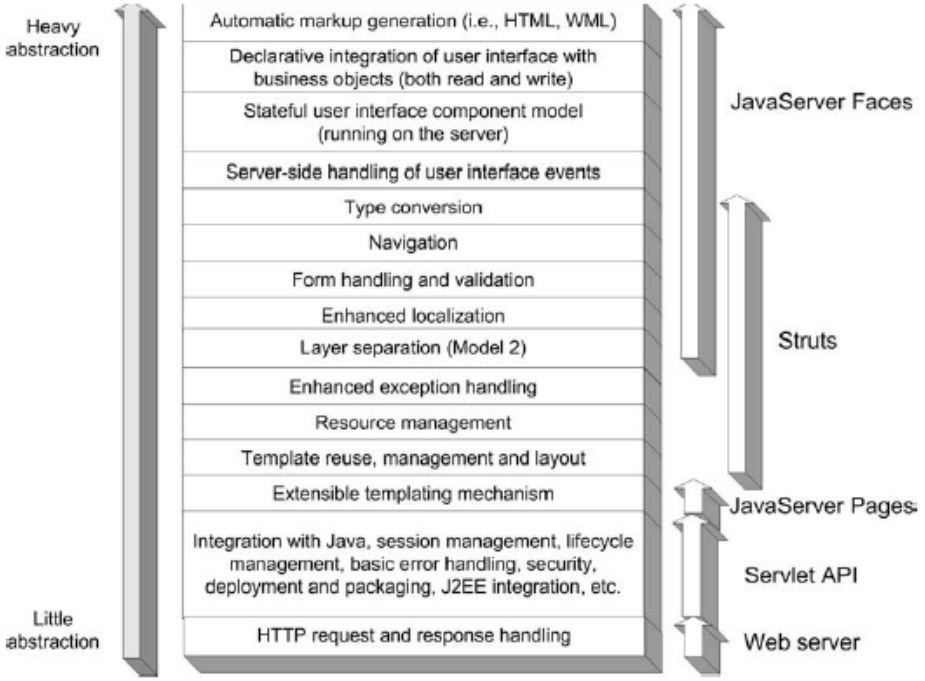
4.1.2 Rozdělení webových frameworků

Webové frameworky rozdělujeme z hlediska míry abstrakce nad protokolem HTTP na požadavkem řízené a na komponentně orientované. Čím větší je úroveň abstrakce, tím méně vývojář pracuje se samotným protokolem (viz obrázek 4.1).[5]

Požadavkem řízené frameworky (Request-Based): Základní prvkem těchto frameworků je požadavek, což přináší vývojáři přesnou kontrolu nad HTTP provozem, neboť pracuje přímo s URL a parametry stránky. Jedná se o poměrně malou abstrakci nad http servlety. Programátor využívá specifických knihoven rámce pro řešení svých specifických úkolů. Použití tohoto typu frameworku je

vhodné pro velmi škálovatelné projekty menšího rozsahu. Mezi nejznámější zástupce patří rámec Struts, Stripes nebo Spring MVC.

Komponentně řízené frameworky (Component-Based): Tyto frameworky umožňují, nebo se alespoň snaží vývojáře odstínit od toho, že tam někde venku je HTTP protokol. Vývojář se tak skoro může zdát, že vyvíjí desktopovou aplikaci. Základním prvkem těchto frameworků je komponenta. Komponenta je nějaká součást (např. webový formulář), která může být znovu použita v rámci programu a tím vývojáři šetří čas. Neznámějšími zástupci této kategorie jsou Apache Wicket, Tapestry, Java Server Faces.[5]



Obrázek 4.1: Abstrakce nad protokolem http

4.1.3 Kritéria výběru webového frameworku

Tato práce není pojednáním o webových rámcích pro Java EE, proto je tato podkapitola, napsána stručně. Při tvorbě této webové aplikace příliš nezáleží na konfiguraci klasického schématu request-response jako spíš na tvorbě komponent, proto při výběru webového frameworku jsem vybíral z komponentně řízených frameworků. Abychom mohli nějakým způsobem porovnat klady a zápory jednotlivých rámců, je potřeba si definovat kritéria, podle kterých budeme porovnávat. Dále je nutné určit, které funkční požadavky z hlediska návrhu a implementace webové aplikace jsou nejdůležitější. Neexistuje univerzální webový framework, vhodný pro jakýkoliv typ aplikace. Proto je nutné vybrat správný framework, protože výběrem nevhodného může dojít ke snížení efektivity vývoje a následné údržby aplikace.

- Jak dlouho trvá vývojářům osvojení si nového frameworku?

- Jak je silná internetová komunita pro daný framework? Kolik vyšlo knih o daném frameworku? Existuje dostatečné množství příkladů? Je volně přístupná dokumentace?
- Na jakých platformách je možné pomocí daného frameworku vyvíjet aplikaci?
- Jakou podporu nabízí framework pro čím dál více používanější mobilní zařízení?
- Jaká je podpora vícejazyčných aplikací v rámci standardizace i18n
- Jaká je zpětná kompatibilita mezi jednotlivými verzemi frameworku?
- Jak perspektivní je další vývoj frameworku? Kdy byly vydaná poslední verze?

Tyto a další kritéria jsou k vidění na obrázku 4.2, kde ke každému kritériu je dána určitá váha, dle důležitosti kritéria.[6]

Weight	Criteria	Struts 2	Spring MVC	Wicket	JSF	Tapestry	Stripes	GWT	Grails	Rails	Flex
10	Developer Productivity	5.00	5.00	5.00	5.00	10.00	5.00	10.00	10.00	10.00	0.00
0	Developer Perception	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	Learning Curve	5.00	5.00	2.50	2.50	2.50	5.00	5.00	5.00	5.00	5.00
5	Project Health	2.50	5.00	5.00	5.00	5.00	2.50	5.00	5.00	5.00	2.50
5	Developer Availability	2.50	5.00	2.50	5.00	5.00	2.50	5.00	2.50	5.00	5.00
0	Job Trends	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	Templating	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	Components	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	Ajax	2.50	5.00	2.50	2.50	2.50	2.50	5.00	2.50	2.50	2.50
5	Plugins or Add-Ons	2.50	0.00	5.00	5.00	2.50	0.00	5.00	5.00	5.00	5.00
10	Scalability	10.00	10.00	5.00	5.00	5.00	10.00	10.00	5.00	5.00	5.00
10	Testing	10.00	10.00	5.00	5.00	10.00	10.00	5.00	10.00	10.00	0.00
0	i18n and l10n	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	Validation	5.00	5.00	5.00	2.50	5.00	5.00	5.00	5.00	5.00	5.00
10	Multi-language Support (Groovy / Scala)	5.00	5.00	10.00	10.00	10.00	10.00	0.00	10.00	0.00	0.00
10	Quality of Documentation/Tutorials	5.00	10.00	5.00	5.00	5.00	10.00	10.00	10.00	10.00	10.00
0	Books Published	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10	REST Support (client and server)	5.00	10.00	5.00	0.00	5.00	5.00	5.00	10.00	10.00	5.00
10	Mobile / iPhone Support	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	5.00
0	Degree of Risk	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	**Weighted Totals**	70	85	67.5	62.5	77.5	77.5	80	90	82.5	50

Obrázek 4.2: Hodnocení frameworků

4.1.4 Apache Wicket

Apache wicket je komponentně řízený Framework pro tvorbu webových aplikací v programovacím jazyce Java. Dle obrázku 2.2 je zřejmé, že Apache Wicket nepatří mezi nejlepší webové frameworky, avšak ne všechny vlastnosti daných frameworků je tak jednoduché změřit. Wicket se v tvorbě komponent podobá Swingu, který je součástí Java Foundation Classes a používá se k tvorbě grafického uživatelského rozhraní. Tudíž Wicket komponenty podporují vše, co je v objektově orientovaném programování samozřejmostí.[7] Dále v tomto frameworku jsou striktně odděleny HTML šablony, které reprezentují vzhled webové aplikace od kódu v programovacím jazyce Java, který zastupuje logiku aplikace. Výhodou tohoto rozdělení je přehlednost a uspořádanost kódu. Propojení těchto dvou vrstev je realizováno pomocí atributu wicket:id u HTML elementů, ke kterým je v plánu vytvořit dynamické chování (to je takové chování, které závisí na vstupních datech) a hodnota těchto atributů je shodná s id parametrem konstruktoru odpovídající komponenty v javovském kódu. To ve výsledku znamená, že HTML je pouze značkovací jazyk a neobsahuje žádné logické struktury, což zvyšuje bezpečnost aplikace. Wicket tedy nevytváří vlastní značkovací jazyk, jako je tomu například u frameworku Tapestry či JavaServer Faces. Tímto striktním rozdělení vzhledu od

logiky lze dosáhnout validních HTML dokumentů dle standardů určených W3C. Tento Framework má v sobě zabudovanou podporu AJAX technologie, která slouží pro asynchronní komunikaci mezi uživatelem a serverem. Hlavní výhodou tohoto řešení je to, že programátor může použít tuto technologii i bez znalosti javascriptu. Další výhodou je, že konfigurace aplikace prostřednictvím XML souborů je umístěna pouze v jednom souboru s minimální nutností konfigurace.

Jako nevýhodou bych uvedl, že HTML šablona je umístěna vedle javovského kódu, což má své opodstatnění (viz výše), ale zároveň znepřehledňuje stromovou strukturu. Toto se však dá odstranit použitím nástroje Maven, kde si programátor může v konfiguračním souboru sám nastavit různé cesty pro HTML a Java soubory. Další nevýhodou je, že stále (v aktuální verzi 6.9.0) nepodporuje elementy a způsoby zápisu meta dat a jiných z HTML5, což je však způsobeno tím, že stále není hotová konečná specifikace tohoto značkovacího jazyka. Avšak za největší nevýhodu považuji, že autoři tohoto frameworku používají anotace běžné v jazyce java jen zřídka. Takže se může stát, že při přechodu na novější verzi, část kódu nebude fungovat, protože některé použitých tříd, či jejich metod nové verzi neexistují. A to i přesto, že v předchozí verzi, tyto třídy, či metody nebyli označeny anotací (konkrétně @Deprecated), která by programátorovi jasně naznačila, že dané třídy či metody se nedoporučuje používat, protože v následující verzi budou pravděpodobně odstraněny.

Pro tento framework jsem se rozhodl proto, jak už bylo výše zmíněno, že Apache Wicket se podobá knihovně Swing, se kterou jsem se již během svých studií setkal, což mi usnadnilo počáteční přechod z vývoje desktopové aplikace v Javě na vývoj webové aplikace.[8,9]

4.2 Ostatní použité technologie

V předcházející pod-kapitole jsme se zabývali výběrem webového frameworku. Avšak při vývoji webové aplikace nestačí pouze základní platforma jakou je webový framework, v našem případě Apache Wicket. Je nutné použít i jiné technologie. Následuje výčet těch nejdůležitějších z nich, které jsou standardem pro vývoj webových aplikací.[10]

HTML

HyperText Markup Language, označovaný zkratkou HTML, je značkovací jazyk pro hypertext (HyperText je text, který obsahuje odkazy na jiné texty). Je hlavním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci dokumentů na Internetu. Aktuálně existuje ve verzi 4.1, ale již existuje částečně hotová specifikace pro HTML5. HTML5 přináší nové značky (např. nové typy formulářů jako je kalendář a jiné), zjednodušený zápisy (např. definice hlavičky dokumentu) a jiné. Dále ještě existuje XHTML, kde X znamená e**X**tensible HTML, který dovoluje HTML použít vlastní elementy. Apache Wicket v aktuální verzi však ještě nepodporuje nové značky HTML5, proto je práce psaná v XHTML. [10]

CSS

Kaskádové styly (neboli CSS) popisují způsob zobrazení dokumentů napsaných pomocí značkovacího jazyku HTML. Tímto je oddělen vzhled od obsahu, což zvyšuje přehlednost kódu. Momentálně se pracuje na třetí verzi (např. zabolené rohy, stíny a jiné). Podpora ze strany prohlížečů je však stále minimální a pro některé nové vlastnosti existuje několik různých verzí zápisů pro různé prohlížeče. Nejpodporovanější vlastnosti z CSS3 se však mohou v aplikaci objevit. [11]

JavaScript

JavaScript je objektově orientovaný skriptovací jazyk, který se často vkládá přímo do HTML kódu stránky. Javascript není Java, společnou mají pouze podobnou syntaxi (oba vycházejí z C/C++) a jméno (které u JavaScriptu bylo vybráno z marketingových důvodů). Tento skript se používá pro určitou interakci mezi klientem a webovou stárnkou(např. animace). Protože se JavaScript spouští na straně klienta, plynou z jeho používání některá bezpečností omezení a jeho používání může být prohlížečem zakázáno. [12]

JQuery

JQuery je knihovna (Framework), která usnadňuje práci s JavaScriptem. Klade důraz na jednoduchost, čitelnost, rychlost. Je multiplatformní (funguje na více operačních systémech) a je dostupná zdarma. Má velmi dobře zpracovanou dokumentaci i návody (převážně v angličtině). JQuery je navíc plně implementován v základu Apache Wicket. [13]

AJAX

AJAX (Asynchronous JavaScript and XML) je obecné označení pro technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovu načítání. Na rozdíl od klasických webových aplikací poskytují uživatelsky příjemnější prostředí, ale vyžadují použití moderních webových prohlížečů. V rámci frameworku Apache Wicket je plně integrován a není tak nutné používat javascript pro volání funkcí z technologie AJAX.

Apache derby

Apache derby je relační databáze vycházející ze strukturovaného dotazovacího jazyka SQL. Implementována v Javě, primární API je JDBC. Je to malá databáze vhodná pro menší projekty, není nutné instalovat server. Původně byla tato práce projektována pro Oracle DB, ale vzhledem k relativně malému počtu záznamů byla jako SŘBD vybrána právě databáze Apache Derby. Vzhledem ke kompatibilitě SQL dotazů, je možnost přejít zpátky na Oracle DB bez větších komplikací.[14]

Maven

Maven je nástroj pro správu, řízení a automatizaci buildů aplikací. Formát jeho projektu je akceptován všemi vývojovými prostředími (IntelliJ IDEA, Eclipse, NetBeans). Poskytuje repositář obsahující víceméně všechny javovské knihovny ve všech verzích, mnoho z nich včetně zdrojových kódů a dokumentace Javadoc. Repositář programátora zbavuje nutnosti starat se o knihovny a jejich požadavky, protože v repositáři má každá knihovna uvedeny i další knihovny, na kterých závisí.

iTextPdf

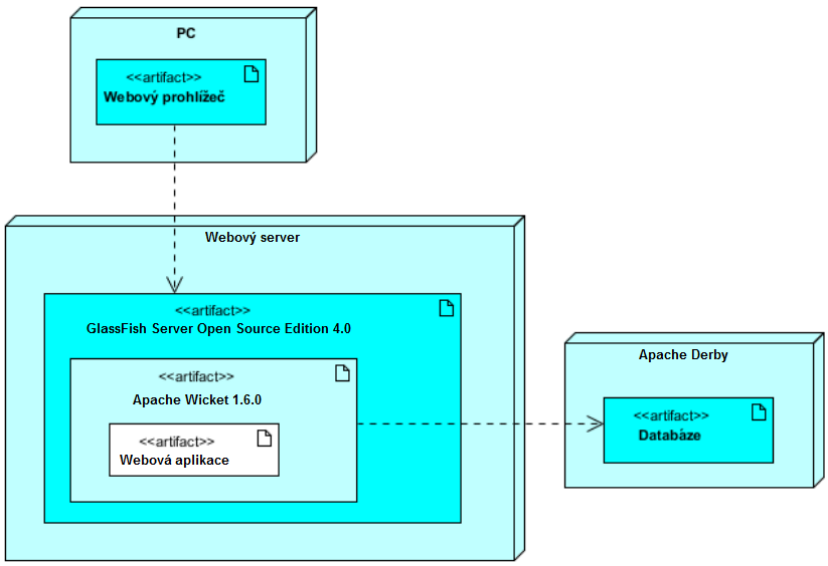
Je vhodné, aby výsledky dotazníku byli přístupné i někomu jinému mimo autora daného dotazníku. Proto je ideální převést výsledky PDF dokumentu. K tomuto účelu byla použita knihovna iTextPDF, která umožňuje vytváření a zpracování PDF dokumentů. Výhodou této knihovny je, že dokáže v omezené míře převést kód HTML a CSS do PDF.

Wicked charts

Wicket charts je knihovna pro vykreslování grafů, vychází z javascriptovské knihovny Highcharts kterou obaluje a přidává k ní rozhraní pro webové aplikace frameworku Apache Wicket. Díky tomu je možné nakonfigurovat grafy přímo v kódu aplikace.

4.3 Diagram nasazení

Diagram nasazení (Deployment Diagram) ukazuje rozložení jednotlivých softwarových komponent na hardwarových zdrojích (uzlech) a jejich spolupráci. Používá se pro specifikaci fyzické architektury systému. Základ fyzické struktury tvoří klientův počítač, server a databáze.



Obrázek 4.3: Diagram nasazení

Klient pomocí webového prohlížeče využívá webový server, na kterém vzhledem k multiplatformnosti Javy může být umístěn jakýkoliv operační systém. Tento server obsahuje aplikační server *GlassFish*. Ten obsahuje Apache Wicket Framework, který je základní frameworkem pro celou aplikaci a zároveň komunikuje s databází.

4.4 Vlastní implementace

4.4.1 NetBeans

NetBeans je vývojové prostředí od Oracle Corporation. Podporuje několik programovacích jazyků (Java, C/C++) a webových technologií (HTML, CSS nebo JavaScript). Samozřejmostí je kontrola syntaxe, doplňování kódu a balíček knihoven pro Javu. Důležitou součástí je modularita, která dovoluje jednoduše rozšířit vývojové prostředí NetBeans o potřebné zásuvné moduly, které mohou vývojové prostředí rozšířit o frameworky, různé testovací funkce a další. V rámci této práce bylo prostředí NetBeans rozšířeno o framework Apache Wicket.

4.4.2 Model

Model reprezentuje data uložená v databázi. Jde o jednoduché Javovské objekty, vytvořených podle návrhového vzoru Table Data Gateway (viz. kaptiola 4.3.2), pomocí kterých lze manipulovat s daty jako s objekty, i když jsou uloženy v relační databázi, tato technika se nazývá Objektově-relační mapování.

```
public class User{
private String login;
public String getLogin() {
    return login;
  }
  public void setLogin(String login) {
    this.login = login;
  }
}
```

Tento kód ukazuje jen část modelové třídy Users. Třída obsahuje proměnné se stejnými názvy, jako mají sloupce v tabulce Users, tyto proměnné jsou definované jako privátní, což znamená, že s nimi mimo třídu nelze nijak manipulovat. Toto se nazývá zapouzdření, abychom mohli s těmito proměnnými manipulovat, je nutné k nim vytvořit veřejné metody (deklarované jako public). V programovacím jazyce Java je zvykem vytvářet tyto metody s názvem get (pro čtení hodnoty) a set (pro nastavení hodnoty) před jménem proměnné. V rámci prostředí NetBeans není nutné tyto takzvané *gettery* a *settery* vypisovat, protože na vyžádání jsou automaticky vygenerovány.

4.4.3 Pohled

Pohled představuje uživatelské rozhraní, pomocí kterého uživatel posílá požadavky řadiči a prohlíží si výsledek. V rámci této webové aplikace je základ tvořen HTML kódem, doplněným o CSS styly. Samotná stránka se může skládat z několika pohledů, tyto tzv. částečné pohledy lze kombinovat v neomezeném počtu.

Není žádoucí, aby pohled pracoval přímo s daty (snižuje to přehlednost). V rámci Apache Wicket je navíc nepohodlné pracovat s daty, či aplikační logikou přímo v pohledu (Je nutné propojit několik technologií). Přípravu dat má na starosti řadič, který pošle data do pohledu právě ve formě modelu, nebo nabízí dynamické vlastnosti, jež lze naplnit libovolnými daty. Pohled už se jen postará o jejich zobrazení.

4.4.4 Řadič

Řadič slouží jako jakási ústřední výkonná jednotka, která se stará o celkové provázání funkčnosti aplikace. Je to prostředník mezi modelem a pohledem. Na základě požadavku z pohledu připraví konkrétní data z modelu a vrátí je opět do (často jiného) pohledu nebo předá řízení do jiného řadiče. V rámci aplikace jsou třídy propojené s HTML stránkami řadiči.

```
UserDAO userDAO = new UserDAO();
HashClass hc = new HashClass(user.getPassword());
```

```
try{
        ((UserSession)Session.get()).setUser(userDAO.getLoginEnter(user.getLogin(),
hc.getPass()));
        SignIn.this.replaceWith(new SignOut(id));
}
catch(Exception e){
        message = "Heslo nebo login je zadano špatně";
}
```

Tato část kódu je ze třídy *SignIn*, kdy po vyplnění přihlašovacího jména a hesla (v rámci pohledu) došlo k odeslání formuláře. Poté dojde k hašování zadaného hesla pomocí hašovací funkce MD5. Hašovací funkce je matematická funkce (resp. algoritmus) pro převod vstupního řetězce do jiného řetězce (Zpětně nepřeveditelného). Poté následuje kontrola, zda uživatel s takovým přihlašovacím jménem a heslem v databázi existuje, pokud ano vymění se pohled a řadič pro přihlašování za odhlašování, v opačném případě je vypsáno chybové hlášení.

Výše uvedené podkapitoly Model, Pohled a Řadič, představují architekturu MVC (Model-View-Controler). Tato architektura dělí aplikaci na tři logické části tak, aby je šlo upravovat samostatně a dopad změn byl na ostatní části co nejmenší. Model reprezentuje data a business logiku aplikace, Pohled zobrazuje uživatelské rozhraní a Řadič má na starosti tok událostí v aplikaci a obecně aplikační logiku.

4.4.5 Optimalizace prohlížečů

Ačkoliv W3C standardizovalo jednotlivé elementy a styly HTML a CSS a definovalo, jak se mají určité prvky v daném případě zobrazovat, tak stále dochází k poněkud vágnímu dodržování těchto standardů ze stran tvůrců prohlížečů. Může například docházet k tomu, že určitá značka či styl je v jednom prohlížeči podporován a v jiném ne, případně daný prvek je v jednom prohlížeči zobrazen tak jak to autor zamýšlel, ale ve druhém přesahuje přes hraniční prvek.

Samotný vývoj této aplikace probíhal v prohlížeči Mozzilla Firefox 22.0, který v kombinaci s doplňkem FireBug nabízí velice kvalitní nástroje kontrolu a ladění HTML elementů a kaskádových stylů. Následně byla aplikace optimalizována pro Google Chrome a Internet Explorer. Tyto prohlížeče zaujímají drtivou většinu tohoto trhu, proto optimalizace nebyla provedena u zbylých prohlížečů.

4.4.6 Bezpečnost

Výsledná aplikace operuje s relativně citlivými údaji a daty. Proto je nutné zvážit míru zabezpečení aplikace.

Samozřejmostí je autentizace a následná autorizace uživatelů, které zabezpečuje přístup do systému pouze uživatelům s jasnou identitou, respektive umožňuje provádět specifické akce pouze oprávněným uživatelům.

Největší nebezpečí tkví v útoku SQL injection a Cross-Site Scripting. SQL injection je technika napadení databázové vrstvy programu vsunutím (odtud „injection“) kódu přes neošetřený

vstup a vykonání vlastního, samozřejmě pozměněného, SQL dotazu. Díky tomuto útoku může dojít ke zkopírování, úpravy, či smazání dat. Avšak v tomto případě díky použití objektově-relačního mapování je možnost úspěchu takovéhoto útoku prakticky vyloučen. Je to dáno tím, že SQL dotazy vytvořené pro běh aplikace jsou parametrizované. Parametrizované dotazy jsou takové, které obsahují neměnné nebo nahraditelné dotazy.

Cross-Site Scripting je metoda narušení WWW stránek využitím bezpečnostních chyb ve skriptech (především neošetřené vstupy). Útočník díky těmto chybám v zabezpečení webové aplikace dokáže do stránek podstrčit svůj vlastní javascriptový kód, což může využít buď pouze k poškození vzhledu stránky, jejímu znefunkčnění anebo dokonce k získávání citlivých údajů návštěvníků stránek, obcházení bezpečnostních prvků aplikace a phishingu. V této aplikaci je jako ochrana použito převedení nebezpečných znaků do HTML entit (např. < za <, > za > atd.).

4.5 Testování aplikace

Testování aplikace je nedílnou součástí souhrnu úkonů, které vedou až k nasazení aplikace do běžného provozu. Pojmem testování je myšleno soubor kroků, které vedou ke zjištění, chybovosti či správné funkcionalitě aplikace, tak jak je předepsáno v její dokumentaci. Testování se většinou provádí od nejmenší možné jednotky (od toho jednotkové testy angl.. Unit test), u které leze předpokládat chybu (např. metoda, ošetření jednoho vstupu), přes segment logicky souvisejícího kódu, až po aplikace jako celku. Podle toho, zda jsou testy prováděny člověkem nebo softwarem, se rozlišuje manuální a automatické testování. Pokud test vyžaduje lidské ohodnocení a úsudek nebo rozličné přístupy, které není třeba zaznamenat a pravidelně opakovat, je vhodnější manuální testování. Pro opakované spouštění velkého množství testů nebo testu s velkým množstvím generovaných dat, stejně jako pro zátěžové testování je dobré použít automatické testy. Pro testování této webové aplikace bylo použito manuálních testů. Ty byli, vytvořeny pomocí testovacích případů a testovacích scénářů.

4.5.1 Testovací případy

Testovací případ (test case) definuje postup, jak otestovat daný požadavek. Pokud hovoříme o požadavcích, máme na mysli funkční a nefunkční požadavky uváděné většinou v dokumentaci. Testovací případ popisuje, jak otestovat požadovanou funkčnost, to znamená, co má být zadáno na vstupu a co lze očekávat na výstup. Existují dva typy testovacích případů a to logický, což je abstraktní popis toho, co se má otestovat a jak se má otestovat (definuje obor a množinu hodnot). A fyzický, což je konkrétní popis toho, co se má otestovat a definice hodnot, které se mají zadat. Pro testování byli použity oba dva typy, nejprve logicky pro definici a popis testovacího scénáře a následně v rámci testovacích scénářů byli použity fyzické testovací případy. Testovací případy naleznete v příloze I.

4.5.2 Testovací scénáře

Testovací scénář je tvořen sadou testovacích případů. Může se jednat o testovací případy, které na sebe navazují a musí být vykonány v přesně uvedeném pořadí. Nebo na pořadí, v jakém jsou jednotlivé testovací případy prováděny, nezáleží. Testovací scénáře naleznete v příloze II.

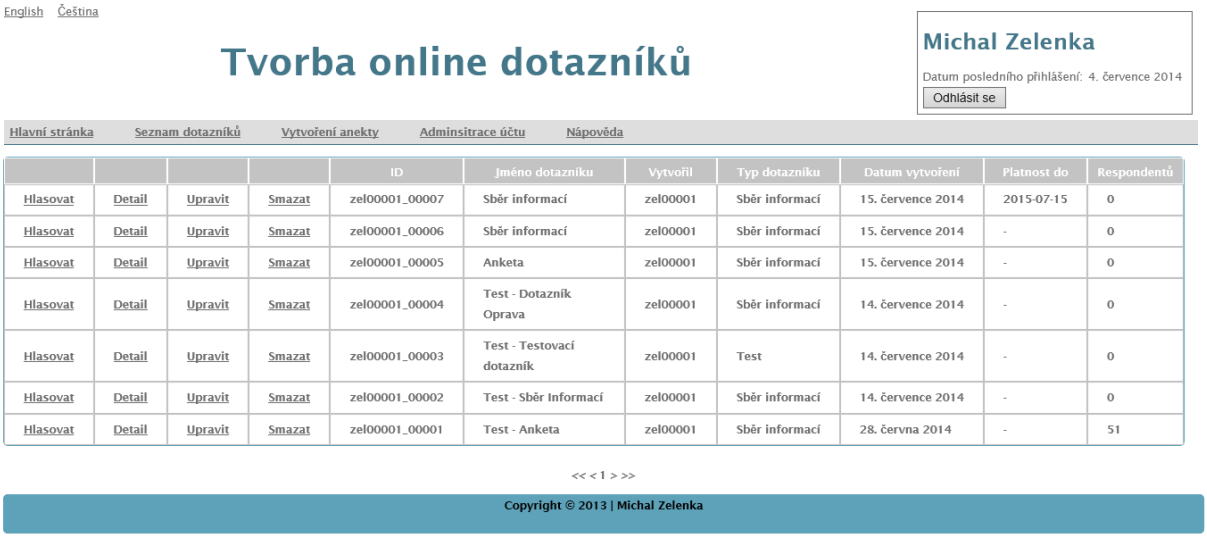
4.6 Ukázky z webové aplikace

V následující sekci jsou prezentované základní obrazovky výsledné aplikace. Po autentizaci uživatele webová aplikace zobrazí hlavní stránku, které dominuje sekce krátkých zpráv. V hlavičce je obsažen nadpis stránky, v jejím levém rohu se nachází dvojice odkazů, jeden pro změnu jazyka stránky na angličtinu a druhý na češtinu. Na pravé straně je umístěn přihlašovací/odhlašovací panel. Hlavička spolu s menu a patičkou zůstává neměnná v celé aplikaci. Nechybí zde ani sekce užitečných odkazů.

Krátké zprávy může vkládat jakýkoliv přihlášen uživatel, takže vidí odkaz pro vložení nového článku. Na hlavní stránce je přístup pouze k posledním pěti zprávám. Vše je vidět na obrázku 4.4.

Obrázek 4.4: Hlavní stránka webové aplikace

Jednou z nejpoužívanějších obrazovek aplikace je samotný seznam dotazníků. Aplikace zde uživatelům vypíše v přehledné tabulce seznam dotazníků. Protože se předpokládá, větší počet dotazníků, je umožněno stránkování po dvaceti položkách. Přihlášeným uživatelům jsou zobrazeny pouze vlastní dotazníky, zatímco ostatním uživatelům jsou zobrazeny dotazníky všech uživatelů. Vše je zachyceno na obrázku 4.5.



Obrázek 4.5:Seznam dotazníků

5 Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat webovou aplikaci, pro online průzkumy s podporou statistického vyhodnocování. Vytyčený cíle této práce se povedlo splnit. Při práci použitý webový Framework Apache Wicket mi jednoznačně ulehčil implementaci aplikace. I tak se vyskytly komplikace, protože vybraný framework neumožňoval vytvoření grafů, pro jejichž vytvoření jsem byl nucen použít javascript a knihovnu pro tvorbu grafů. Přesto v tomto frameworku vidím přínos i do budoucna v rámci správy a rozšiřování funkcionality této aplikace. Pro budoucí vývoj této aplikace by bylo vhodné ji rozšířit o uživateli volený vzhled, což by mohlo napomoct atraktivitě dotazníků pro respondenty i tazatele. Jako největší výzvu pro další vývoj této aplikace však považuji rozšíření identifikace respondentů podle přihlašovacích údajů dané stránky. Takovéto rozhraní by však vzhledem k rozšířenosti sociálních síti mělo být vytvořeno spíše dle specifikací daných sítí, než na základě vlastního návrhu.

Seznam obrázků

Seznam tabulek

Použitá literatura

[1] Web application in: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, poslední revize 22.6.2013 [cit. 2013-6-30]. <http://en.wikipedia.org/wiki/Web_application>

[2] FOWLER, Martin. Patterns of Enterprise Application Architecture. 1st printing, 2002. Boston (Massachusetts): Addison-Wesley. 560 s. ISBN 0-321-12742-0.

[3] Lazy loading in: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, poslední revize 23.6.2013 [cit. 2013-7-1]. <http://en.wikipedia.org/wiki/Lazy_loading>

[4] Design Patterns: Data Access Object in: Oracle[online]. Santa Clara (CA) : Oracle Corporation, 1977-, [cit. 2013-7-1]. < http://www.oracle.com/technetwork/java/dao-138818.html>

[5] MANN, Kito. JavaSever Faces In Action. 1st printing, 2005. New Delhi (IN): Dreamtech Press. 744 s. ISBN 1-932394-11-7.

[6] JVM Web Framework Matrix[online]. 2010, poslední revize 25.3.2013 [cit. 2013-5-6]. <https://spreadsheets.google.com/spreadsheet/pub?hl=en&key=0AtkkDCT2WDMXdC1HOEtnUHpCejJMbUhGeGJWUmh5dVE&hl=en&gid=0>

[7] Swing (Java) in: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, poslední revize 31.5.2013 [cit. 2013-6-9]. <http://en.wikipedia.org/wiki/Swing_java>

[8] DASHORST, Martin a HILLENIUS, Eelco. *Wicket In Action*. 1st printing, 2008. Greenwich (Connecticut): Manning publictation Co. 364 s. ISBN 1-932394-98-2.

[9] VAYNBERG, Igor. Apache Wicket Cookbook. 1st printing 2011. Birmingham (UK): Packt Publishing. 312 s. ISBN 1849511608.

[10] Dušan Janovský. HTML příručka [Online]. 2001-, poslední revize 6.12.2012 [cit. 2013-7-12] <http://www.jakpsatweb.cz/html/formulare.html>

[11] Dušan Janovský. CSS kaskádové styly [Online]. 2001-, poslední revize 6.12.2012 [cit. 2013-7-12] <http://www.jakpsatweb.cz/css/css-pozicovani.html>

[12] Dušan Janovský. Javascript, návody na použití jazyka [Online]. 2001-, poslední revize 6.12.2012 [cit. 2013-7-12] <http://www.jakpsatweb.cz/javascript/funkce.html>

[13] JQuery[online]. 2006-, [cit. 2013-7-14]. < http://api.jquery.com>

[14] Apache Derby[online]. 2004-, poslední revize 17.4.2014, [cit. 2013-7-14].

< http://db.apache.org/derby/>

Přílohy

CD obsahuje složky:

1. Aplikace – zdrojové kódy vypracované aplikace
2. WebAplication.war – soubor s vlastní aplikací
3. Nástroje – nástroje použité při vývoji aplikace
4. Příloha_I_BP_ZEL0021.pdf – dokument, příloha k bakalářské práci
5. Příloha_II_BP_ZEL0021.pdf – dokument, příloha k bakalářské práci obsahující testovací scénáře
6. Create.sql – dokument, obsahující příkazy pro vytvoření tabulek k databázi
7. Insert.sql – dokument, obsahující příkazy pro vložení dat do tabulek v databázi